1- The recurrence relation for the algorithm;

$T(n) = 2T(n-1) + O(1)$

$T(1) = 2T(0) + O(1)$

$T(2) = 4T(0) + 2O(1)$

⋮

$T(n) = 2^n T(0) + (2^n - 1)$, since $T(0)$ is $O(1)$ (current set is empty and no discount) the time complexity of this function is $O(2^n)$.

2- On best - avg - worst case, we need to calculate the cost for all permutations because we don't know whether there are cheaper combination.

The algorithm computes the total cost for each combination to fill the matrix so the time complexity is $O(n!)$

3- Like the previous algorithm, we need to calculate the cost of all possible sequences to find the minimum energy consumption. There are $n!$ sequence. $f(n) \in O(n!)$

5-

$T(n) = 2T(n/2) + O(1)$

$T(n) = 2T(n/2) + 1$

$= 4T(n/4) + 3$

$= 8T(n/8) + 7$

(k) ⋮

$2^k T(n/2^k) + 2^k - 1$ for $2^k = n$, $T(n) = nT(1) + n - 1 \in O(n)$.

4- Algorithm sorts the coins and calls them from biggest to smallest by checking whether the current coin should be called (coin[i] ≤ change-sum) until the sum reaches change. The time complexity is $O(n \times n)$ since 2 for loops.